CTR 预估模型的进化之路

导语

笔者对各大厂商CTR预估模型的优缺点进行对比，并结合自身的使用和理解，梳理出一条CTR预估模型的发展脉络，希望帮助到有需要的同学。

0. 提纲

1. 背景
2. **LR 海量高纬离散特征** (广点通精排)
3. **GBDT 少量低纬连续特征** (Yahoo & Bing)
4. **GBDT+LR** (FaceBook)
5. **FM+DNN** (百度凤巢)
6. **MLR** (阿里妈妈)
7. **FTRL_Proximal** (Google)

1. 背景

众所周知，广告平台的最终目标是追求收益最大化，以 CPC 广告为例，平台收益既与 CPC 单价有关，又与预测 CTR 有关。在排序的时候，CPC 可以认为是一个确定的值，所以这里的关键是预测用户的点击率 pCTR。

$$MaxValue = f(流量_1, ..., 流量_n) = \sum_{i=1}^{n} 流量_i \times (pCTR_i)^w \times CPC_i \times 扶持力度_i$$

(指数项 w 是一个调节因子，用于平衡用户体验和收入。扶持力度用于调节各个广告渠道)

互联网公司根据各自业务的特点，研发出了各种各样的 CTR 预估模型及其变种，本文尝试在众多流派和分支中梳理出一条 CTR 预估模型的发展脉络。

2. LR 海量高纬离散特征 (广点通精排)

LR（逻辑回归）1可以称之上是 CTR 预估模型的开山鼻祖，也是工业界使用最为广泛的 CTR 预估模型。LR 是广义线性模型，与传统线性模型相比，LR 使用了 Logit 变换将函数值映射到 0~1 区间，映射后的函数值就是 CTR 的预估值。

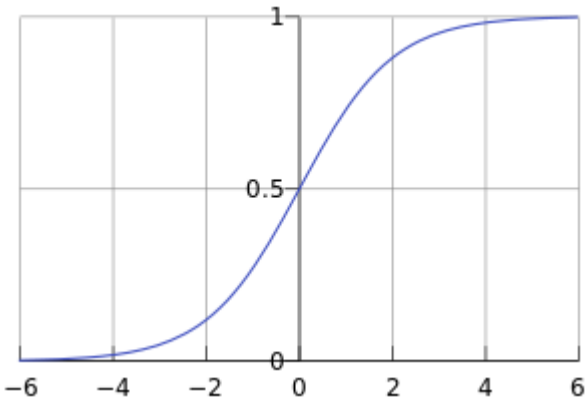
LR 利用了 Logistic 函数，函数形式为：

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$

对于线性边界，边界形式如下：

$$\theta_0 + \theta_1 x_1 + ,\ldots, + \theta_n x_n = \sum_{i=1}^{n} \theta_i x_i = \theta^T x$$

Logistic 函数在有个很漂亮的"S"形，如下图所示：



构造 log 损失函数，用梯度下降法求最小值，得到参数向量θ：

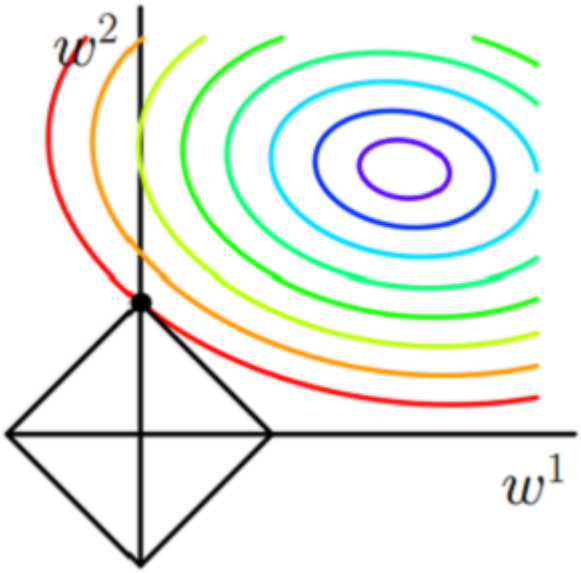
$$J(\theta) = \frac{1}{m}\sum_{i=1}^{n} Cost(h_\theta(x_i), y_i) = -\frac{1}{m}\left[\sum_{i=1}^{n} y_i \log h_\theta(x_i) + (1-y_i)\log(1-h_\theta(x_i))\right]$$

2.1 正则化

为了防止过拟合，通常会在损失函数后面增加惩罚项 L1 正则或者 L2 正则：

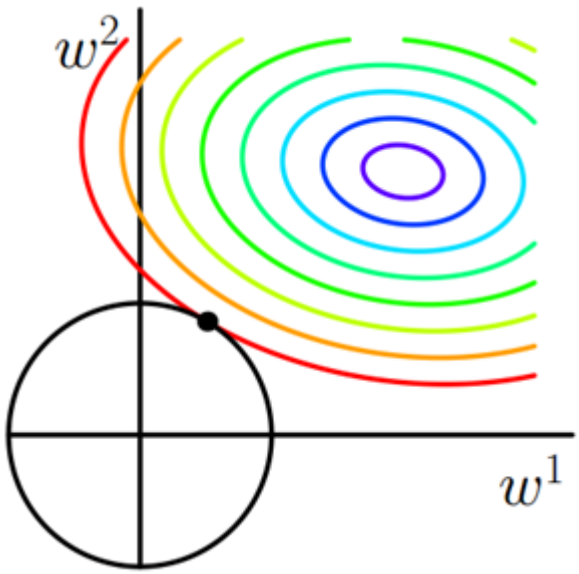
- **L1** 正则化是指权值向量 w 中各个元素的绝对值之和，通常表示为||w||1；
- **L2** 正则化是指权值向量 w 中各个元素的平方和然后再求平方根，通常表示为||w||2。

其中，**L1** 正则可以产生**稀疏性**，即让模型部分特征的系数为 0。这样做有几个好处：首先可以让模型简单，**防止过拟合**；还能**选择有效特征**，提高性能。



如上图：最优解出现在损失函数的等值线和约束函数 L1 相切的地方，即凸点，而菱形的凸点往往出现在坐标轴上（系数 w1 或 w2 为 0），最终产生了稀疏性。

L2 正则通过构造一个所有参数都比较小的模型，**防止过拟合**。但 L2 正则不具有稀疏性，原因如下图，约束函数 L2 在二维平面下为一个圆，与等值线相切在坐标轴的可能性就小了很多。



2.2 离散化

LR 处理离散特征可以得心应手，但处理连续特征的时候需要进行离散化。通常连续特征会包含：大量的反馈 CTR 特征、表示语义相似的值特征、年龄价格等属性特征。

以年龄为例，可以用**业务知识分桶**：用小学、初中、高中、大学、工作的平均年龄区间做分桶；也可以通过**统计量分桶**，使各个分桶内的数据均匀分布。

反馈 CTR 特征的离散化，一般通过**统计量分桶**，但在桶的边界往往会出现突变的问题，比如两个桶分别为 0.013~0.015、0.015~0.018，在边界左右的值 0.01499 和 0.01501 会带来完全不同的效果。（这里给 GBDT LR 埋下一个伏笔）。

2.3 特征组合

LR 由于是线性模型，不能自动进行非线性变换，需要大量的人工特征组合。以 id 类特征为例，用户 id 往往有上亿维(one-hot)，广告 id 往往有上百万维，特征组合会产生维度爆炸。

广告特征里往往有三类维度(a,u,c)，分别是广告类特征、用户类特征、上下文类特征。这三类特征内部两两组合、三三组合，外部再两两组合、三三组合就产生了无限多种可能性。所以**在 CTR 预估模型的早期，主要工作就是在做人工特征工程**。人工特征工程不但极为繁琐，还需要大量的领域知识和试错。

2.4 优缺点

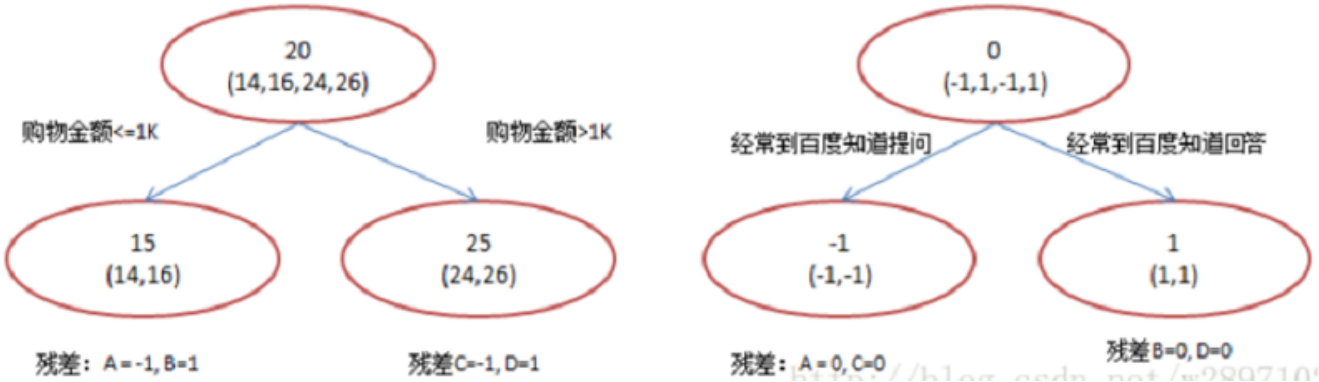
优点：由于 LR 模型简单，训练时便于并行化，在预测时只需要对特征进行线性加权，所以**性能比较好**，往往适合处理**海量 id 类特征**，用 id 类特征有一个很重要的好处，就是**防止信息损失**（相对于范化的 CTR 特征），对于头部资源会有更细致的描述。

缺点：LR 的缺点也很明显，首先对连续特征的处理需要先进行**离散化**，如上文所说，人工分桶的方式会引入多种问题。另外 LR 需要进行**人工特征组合**，这就需要开发者有非常丰富的领域经验，才能不走弯路。这样的模型迁移起来比较困难，换一个领域又需要重新进行大量的特征工程。

3. GBDT 少量低纬连续特征 (Yahoo & Bing)

GBDT（Gradient Boosting Decision Tree）2是一种典型的基于回归树的 boosting 算法。学习 GBDT，只需要理解两方面：

- **a. 梯度提升（Gradient Boosting）**：每次建树是在之前建树损失函数的梯度下降方向上进行优化，因为梯度方向(求导) 是函数变化最陡的方向。不断优化之前的弱分类器，得到更强的分类器。每一棵树学的是之前所有树结论和的残差。
- **b. 回归树（Regression Tree）**：注意，这里使用的是回归树而非决策树，通过最小化 log 损失函数找到最靠谱的分支，直到叶子节点上所有值唯一 (残差为 0)，或者达到预设条件(树的深度)。若叶子节点上的值不唯一，则以该节点上的平均值作为预测值。



核心问题 1：回归树怎么才能越来越好？

最直观的想法，如果前一轮有分错的样本，那边在后面新的分支只需提高这些分错样本的权重，让没学好的地方多学一学。这种方法在数学上可以用**残差**来很好的解决，比如上图中，第一轮训练后残差向量为(-1, 1, -1, 1)，第二轮训练就是为了消除残差，即这些分错的样本，当残差为 0 或者达到停止条件才停止。

那么哪里体现了**Gradient**呢？其实回到第一棵树结束时想一想，无论此时的 cost function 是什么，是均方差还是均差，只要它以误差作为衡量标准，残差向量(-1, 1, -1, 1) 都是它的全局最优方向，这就是 Gradient。

核心问题 2：如何将多个弱分类器组合成一个强分类器？

通过加大分类误差率较小的弱分类器的权重，通过**多棵权重不同的树**（能者多劳）进行打分，最终输出回归预测值。

3.1 特征工程

由于 GBDT 不善于处理大量 id 类离散特征（后文会讲到），但是却善于处理连续的特征，一般的做法是用**各种 CTR 反馈特征来做交叉**，来范化的表达信息。在这种情况下，**信息会大量存在于动态特征中，而少量存在于模型中**（对比 LR，信息几乎都存在于模型中）。下图是作者为搜索广告的 GBDT 模型设计的特征，读者可供参考。

3.2 优缺点

优点：我们可以把树的生成过程理解成**自动进行多维度的特征组合**的过程，从根结点到叶子节点上的整个路径(多个特征值判断)，才能最终决定一棵树的预测值。另外，对于**连续型特征**的处理，GBDT 可以拆分出一个临界阈值，比如大于 0.027 走左子树，小于等于 0.027（或者 default 值）走右子树，这样很好的规避了人工离散化的问题。

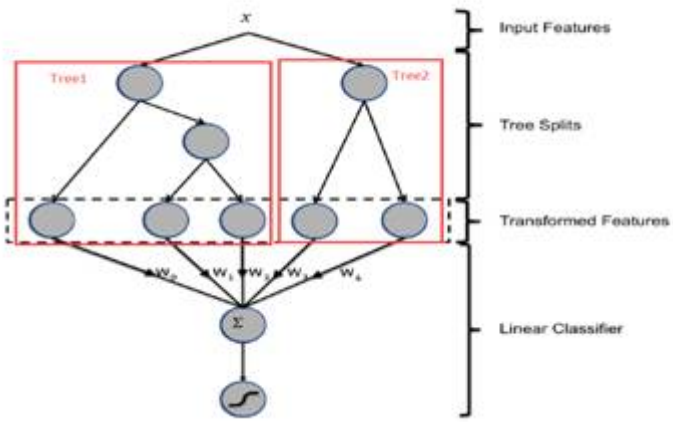
缺点：对于**海量的 id 类特征**，GBDT 由于树的深度和棵树限制（防止过拟合），不能有效的存储；另外海量特征在也会存在性能瓶颈，经笔者测试，当 GBDT 的 one hot 特征大于 10 万维时，就必须做分布式的训练才能保证不爆内存。所以 GBDT 通常配合少量的反馈 CTR 特征来表达，这样虽然具有一定的范化能力，但是同时会有**信息损失**，对于头部资源不能有效的表达。

4. GBDT LR (FaceBook)

Facebook 在 2014 年发表文章介绍了通过 GBDT 解决 LR 的特征组合问题3，其主要实现原理是：

训练时，GBDT 建树的过程相当于自动进行的特征组合和离散化，然后从根结点到叶子节点的这条路径就可以看成是不同特征进行的特征组合，用叶子节点可以唯一的表示这条路径，并作为一个离散特征传入 LR 进行**二次训练**。

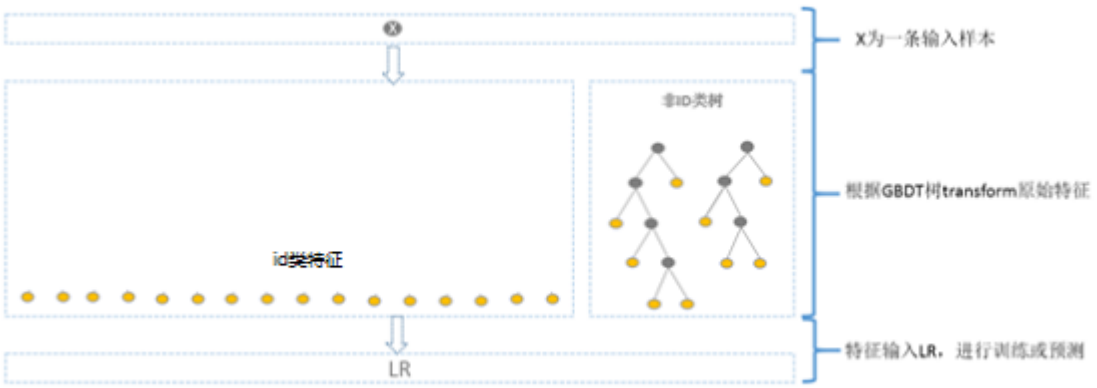
预测时，会先走 GBDT 的每棵树，得到某个叶子节点对应的一个离散特征(即一组特征组合)，然后把该特征以 one-hot 形式传入 LR 进行线性加权预测。



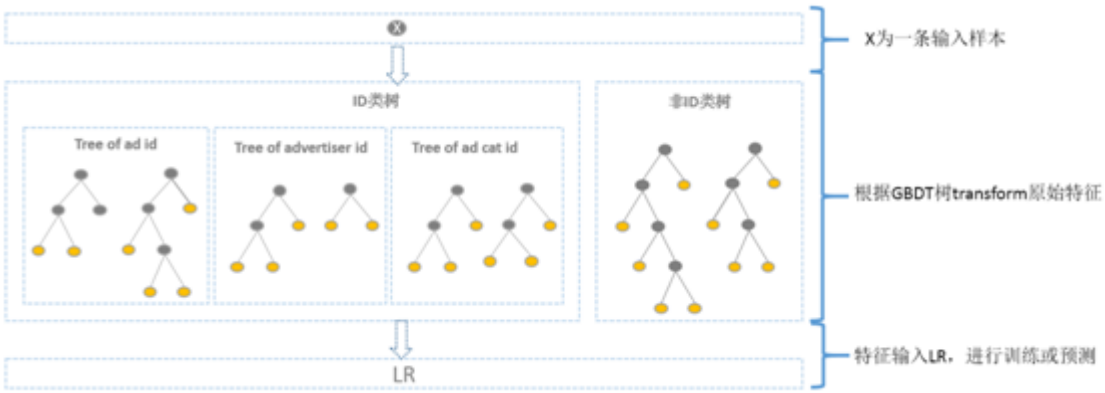
4.1 改进

Facebook 的方案在实际使用中，发现并不可行，因为广告系统往往存在上亿维的 id 类特征(用户 guid10 亿维，广告 aid 上百万维)，而 GBDT 由于树的深度和棵树的限制，无法存储这么多 id 类特征，导致信息的损失。有如下改进方案供读者参考：

方案一：GBDT 训练除 id 类特征以外的所有特征，其他 id 类特征在 LR 阶段再加入。这样的好处很明显，既利用了 GBDT 对连续特征的自动离散化和特征组合，同时 LR 又有效利用了 id 类离散特征，防止信息损失。



方案二：GBDT 分别训练 id 类树和非 id 类树，并把组合特征传入 LR 进行二次训练。对于 id 类树可以有效保留头部资源的信息不受损失；对于非 id 类树，长尾资源可以利用其范化信息（反馈 CTR 等）。但这样做有一个缺点是，介于头部资源和长尾资源中间的一部分资源，其有效信息即包含在范化信息(反馈 CTR) 中，又包含在 id 类特征中，而 GBDT 的非 id 类树只存的下头部的资源信息，所以还是会有部分信息损失。



4.2 优缺点

优点：GBDT 可以自动进行特征组合和离散化，LR 可以有效利用海量 **id** 类离散特征，保持信息的完整性。

缺点：LR 预测的时候需要等待 GBDT 的输出，一方面 GBDT在线预测慢于单 LR，另一方面 GBDT 目前不支持在线算法，只能以离线方式进行更新。

5. FM DNN (百度凤巢)

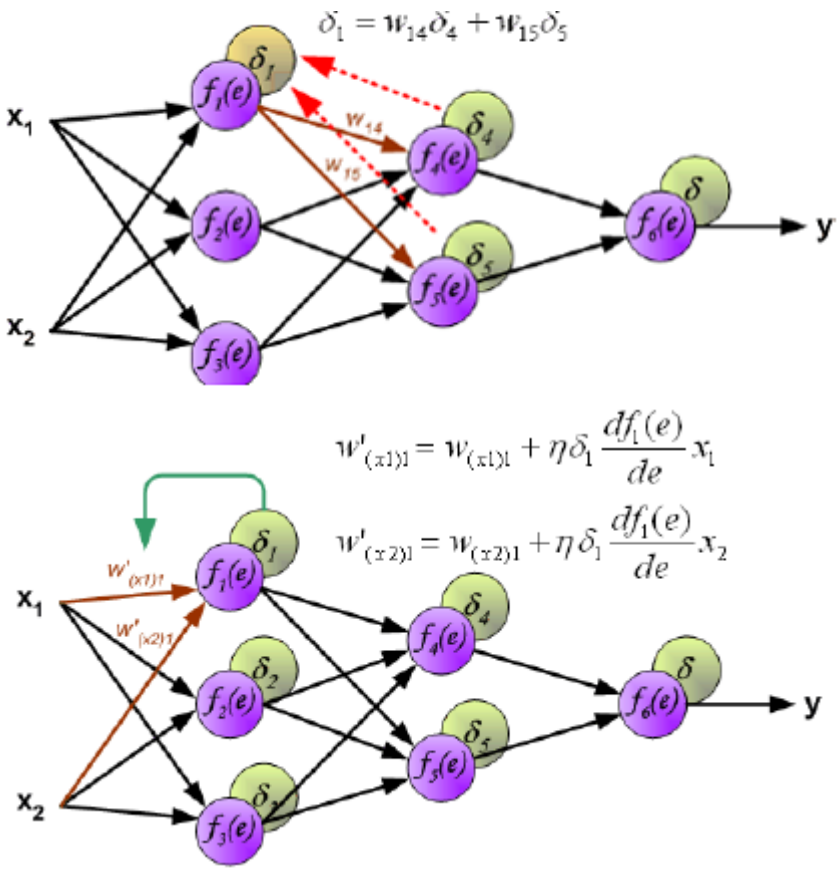
随着深度学习的逐渐成熟，越来越多的人希望把深度学习引入 CTR 预估领域，然而由于广告系统包含海量 id 类离散特征，如果全用 one-hot 表示，会产生维度爆炸，DNN 不支持这么多维的特征。目前工业界方案是 FNN4，即用 FM 做 Embedding，DNN 做训练。

FM（factorization machines）5的目标函数如下：

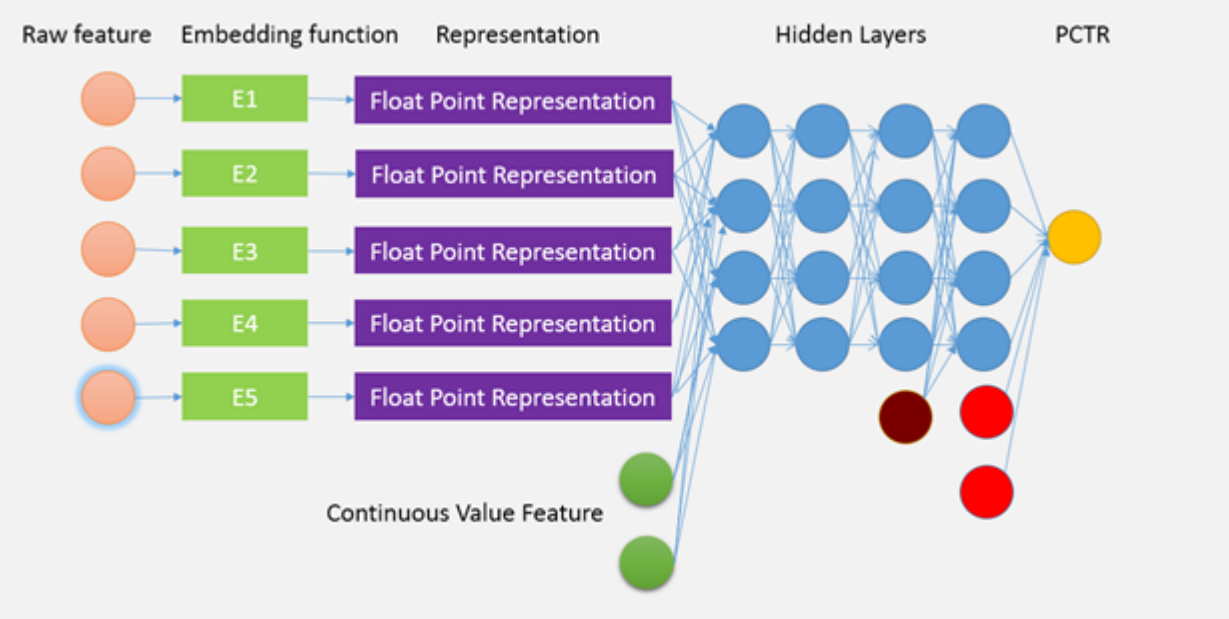
$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j$$

可以看到 FM 的前半部分可以理解成是 LR，后半部分可以理解成是特征交叉(2 阶 FM 只支持 2 维特征交叉)。

DNN(Deep Neural Networks) 6模型往往具有多个隐层，通过 Relu、tanh 等激活函数做非线性变换，残差会反向传播，并通过随机梯度下降来更新权值向量，最终预测时通过 sigmoid 函数做归一化输出（二分类用 sigmoid 做归一化；多分类用 softmax 做归一化）



FM DNN 的具体做法是：



对于**离散特征**，先找到其对应的 category field，并用 FM 做 embedding，把该 category field 下的所有特征分别投影到这个低维空间中。以用户 query 为例，如果用 one-hot 可能有数十亿维，但是如果我们用 FM 编码，就可以把所有的 query 都映射到一个 200 维的向量连续空间里，这就大大缩小了 DNN 模型的输入。

而对于**连续特征**，由于其特征维度本来就不多，可以和 FM 的输出一同输入到 DNN 模型里进行训练。

6.1 优缺点

优点：FM 可以**自动进行特征组合**，并能把同一 category field 下的海量离散特征**投影到一个低纬的向量空间里**，大大减少了 DNN 的输入。而 DNN 不但可以做**非线性变换**，还可以做**特征提取**。

缺点：由于 2 阶**FM 只支持 2 维的特征交叉**（考虑性能的因素常用 2 阶 FM），所以不能像 GBDT 那样做到 10 维的特征组合。另外 DNN 模型出于**调参复杂**和**性能不高**的原因，并不适用于中小型业务。所以在工业界使用的不多。

6. MLR (阿里妈妈)

前一阵子阿里巴巴的广告部阿里妈妈公布了其 MLR(mixed logistic regression) 模型 7，MLR 是 LR 的一个改进， 它采用分而治之的思想，用分片线性的模式来拟合高维空间的非线性分类面。

简单来说，MLR 就是**聚类 LR**，先对样本空间进行分割，这里有一个超参数 m，用来代表分片的个数，当 m=1 时自动沦为普通的 LR；当 m 越大，拟合能力越强；当然随着 m 增大，其所需要的训练样本数也不断增大，性能损耗也不断增加。阿里的实验表明，当 m=12 时，表现最好。

$$x = (x_1, x_2), \quad x_1 \in R^{d_1}, x_2 \in R^{d_2}, x \in R^{d_1+d_2}$$

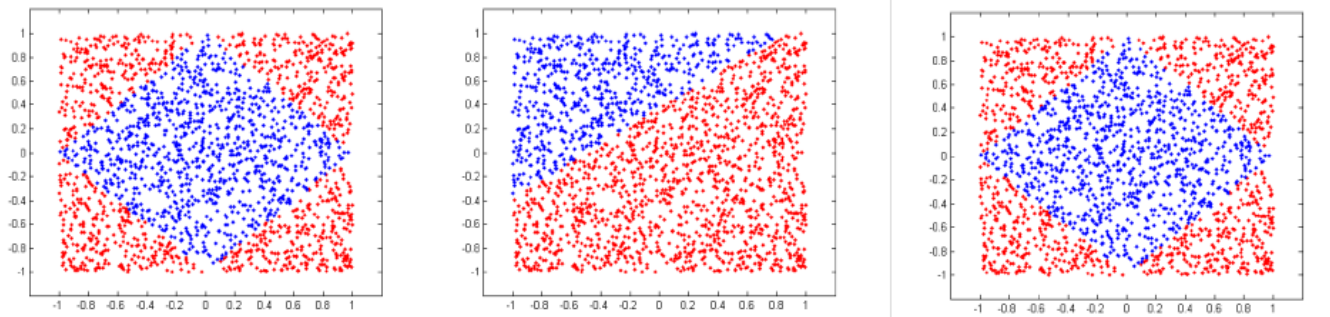
$$f(x;\theta) = \sum_{j=1}^{m} \pi_j(x_1) \cdot \eta_j(x_2) \longleftarrow \text{结构先验/正则}$$

$$= \sum_{j=1}^{m} \frac{exp(\mu_j * x_1)}{\sum_{i=1}^{m} exp(\mu_i * x_1)} \cdot \frac{1}{1 + exp(-w_j * x_2)}$$

X1：聚类参数
决定空间的划分

X2：分类参数
决定空间内的预测

上图公式可以看到，左边聚类部分用 **softmax** 做分区函数并决定样本空间的划分，右预测部分用 **sigmoid** 做拟合函数，并决定空间内的预测。



A) Training dataset　　B) LR model　　C) LS-PLM model

上图可以看到，对于这种高纬空间的非线性分类面，用 MLR 的效果比 LR 好。

6.1 优缺点

优点：MLR 通过先验知识**对样本空间的划分**可以有效提升 LR 对**非线性的拟合能力**，比较适合于电商场景，如 3C 类和服装类不需要分别训练各自不同的 LR 模型，学生人群和上班族也不需要单独训练各自的 LR，在一个 LR 模型中可以搞定。模型的**迁移能力**比较强。

缺点：MLR 中超参数 **m**需要人工去调，另外还是有 LR 共性的缺点，如**需要人工特征组合**和**人工离散化分桶**等。

7. FTRL_Proximal (Google)

对于 LR 静态特征这种模型，信息主要存储在模型中（相比 GBDT 动态特征，信息既存储在模型中又存储在动态特征里），所以为了让模型更加快速的适应线上数据的变化，google 率先把在线算法 FTRL 引入 CTR 预估模型 8。

online 算法其实并不复杂，batch 算法需要遍历所有样本才能进行一轮参数的迭代求解(如随机梯度下降)，而 online 算法可以每取一个训练样本，就对参数进行一次更新，大大提升了效率。但这样做也引入了一个问题，模型迭代不是沿着全局的梯度下降，而是沿着某个样本的梯度进行下降，这样即使用 L1 正则也不一定能达到稀疏解。

FTRL 是在传统的在线算法如 FOBOS 和 RDA 的基础上做了优化，它采用和 **RDA** 一样的 **L1** 正则，同时和 **FOBOS** 一样会限制每次更新的距离不能太远。所以它在稀疏性和精确度上要优于 RDA 和 FOBOS9。

$$
\begin{array}{rrlll}
\text{FOBOS} & x_{t+1} & = \arg\min_x & g_{1:t} \cdot x & + \ \phi_{1:t-1} \cdot x + \Psi(x) \quad + \frac{1}{2} \sum_{s=1}^{t} \|Q_s^{\frac{1}{2}}(x - x_s)\|_2^2 \\
\text{AOGD} & x_{t+1} & = \arg\min_x & g_{1:t} \cdot x & + \ \phi_{1:t-1} \cdot x + \Psi(x) \quad + \frac{1}{2} \sum_{s=1}^{t} \|Q_s^{\frac{1}{2}}(x - 0)\|_2^2 \\
\text{RDA} & x_{t+1} & = \arg\min_x & g_{1:t} \cdot x & + \ t\Psi(x) \quad + \frac{1}{2} \sum_{s=1}^{t} \|Q_s^{\frac{1}{2}}(x - 0)\|_2^2 \\
\text{FTRL-Proximal} & x_{t+1} & = \arg\min_x & g_{1:t} \cdot x & + \ t\Psi(x) \quad + \frac{1}{2} \sum_{s=1}^{t} \|Q_s^{\frac{1}{2}}(x - x_s)\|_2^2
\end{array}
$$

另外 FTRL 采用**Per-Coordinate Learning Rates**，意思是对于某个特征的完整性比较低的时候，会动态加大学习率，而对于特征完整性高的，会减小学习率，让特征权重慢慢学。学习率的计算公式如下，即步长等于历次梯度的平方和的开方的倒数：

$$
\eta_{t,i} = \frac{\alpha}{\beta + \sqrt{\sum_{s=1}^{t} g_{s,i}^2}},
$$

7.1 优缺点

优点：FTRL 可以在线训练，这样使 LR 存储于模型中的信息可以得到快速的更新。另外 FTRL 也具有不错的稀疏性和精确度，可以减少内存占用防止过拟合。最后 FTRL_Proximal 还支持动态的学习率，可以对不同完整性的特征采用不同的步长进行梯度下降。

缺点：FTRL 同样具有 LR 的缺点，需要人工特征工程和人工离散化分桶。

引用

1. Chapelle O, Manavoglu E, Rosales R. Simple and scalable response prediction for display advertising

2. J. H. Friedman. Greedy function approximation: A gradient boosting machine. Annals of Statistics 29, 1189–1232. 2001.

3. Practical lessons from predicting clicks on ads at facebook. He X, Pan J, Jin O, et al.

4. Deep Learning over Multi-field Categorical Data – A Case Study on User Response Prediction. Weinan Zhang.

5. Factorization Machines. Steffen Rendle.

6. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The shared views of four research group. G. E. Hinton et al

7. Learning Piece-wise Linear Models from Large Scale Data for Ad Click Prediction

8. McMahan H B, Holt G, Sculley D, et al. Ad click prediction: a view from the trenchesC//Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2013: 1222-1230.

9. H. B. McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and L1 regularization. In AISTATS, 2011

编辑于 2017-07-31